Vision, Modeling, and Visualization (2012)
M. Goesele, T. Grosch, B. Preim, H. Theisel, and K. Toennies (Eds.)

GPU-accelerated Interactive Material Aging

Tobias Günther Kai Rohmer Thorsten Grosch

Computational Visualistics group, University of Magdeburg, Germany

Abstract

A photorealistic appearance of a 3D scene is required in many applications today. Thereby, one vital aspect is the usage of realistic materials, for which a broad variety of reflectance models is available. When directly employing those models, surfaces always look new, which contrasts strongly the real objects surrounding us as they have undergone diverse kinds of aging processes. The literature already proposes a set of viable methods to simulate different aging phenomena, but all of them are computationally expensive and can thus only be computed off-line. Therefore, this paper presents the first interactive, GPU-accelerated method to simulate material aging in a given scene. Thereby, our approach allows artists to precisely control the course of the aging process. Our particle-based method is capable to reproduce the most common deterioration phenomena in a few seconds, including plausible dirt bleeding, flow effects, corrosion and patina.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction

The faithful reproduction of realistic appearances is one of the ultimate goals computer graphics researchers have been pursuing for decades. Thereby, a key aspect is the resemblance to natural phenomena, among those the weathering and aging of materials over time. Content artists spend much time meticulously facilitating the appearance of 3D objects by editing their textures, whereby each inaccuracy can cause implausibility and quickly break the immersion. For example, a repositioned object undergoes different processes if its weathering side changed or is suddenly blocked, which can also render the content artist's work void if a scene designer decides to rearrange the objects. Not only to favor plausibility of effects caused by interacting objects and to lower production costs, fast simulations of weathering phenomena are vitally needed. Furthermore, robust simulations can contribute to ever-changing environments, thereby taking virtual reality to a new experience. Several methods exist to model the simulation of the chemical, biological and mechanical processes involved here, but run in the order of minutes or hours, thus are far from reaching interactive frame rates, yet.

In this paper, we demonstrate the – to the best of our knowledge – first simulation of aging processes at interactive frame rates, by harnessing the processing power of modern GPU architectures. Thereby, our solution enables artists to steer and design the aging process to eventually generate textures encoding surface properties, e.g. precipitate, normal and height. These are universally applicable for content creation, e.g. in games or production rendering. We do not aim for physical, chemical or biological correctness – instead we show how simple rules and an intuitive user interface can be used to quickly simulate many of the most common aging effects. We exploit the latest GPU features to display the visibly aging scene in an interactive process, whereas an extensively weathered look can be attained in a few seconds. Fig. 1 shows two aged scenes simulated interactively.

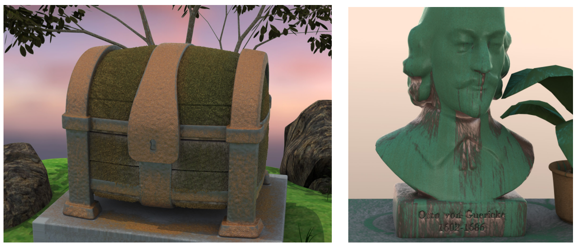


Figure 1: *Interactively aged Chest and Otto scene, exported and rendered in a DCC tool.*

DOI: 10.2312/PE/VMV/VMV12/063-070

2. Previous Work

Material aging processes can be categorized into chemical, mechanical, and biological processes [MG08]. Chemical processes include the temporal change of metal, e.g. corrosion introducing rust [MDG01], or the formation of a patina layer on copper [DH96]. Mechanical processes alter the shape of the object, including the deformation due to cracks [HTK98, HTK00, GC01], peeling [WNH97, PPD02], and man-made influences like scratches [BPMG04] and impacts [PPD01] on the surface. Examples of biological processes are organic lichen growth [DGA04], mould and decay as well as wrinkles in fruits [KRB11]. Many aging processes, like erosion and weathered stone [DEJ*99], can be characterized as a combination of multiple processes [MG08]. As a simulation type, statistical methods can be used [WTL*06], as well as captured photographs over time [GTR*06], or particle simulations. Particles are especially expedient to describe the flow of dirt along the surface [DPH96]. A more general particle-based approach was presented by Chen et al. [CXW*05], allowing the simulation of a wider range of aging phenomena, including dirt bleeding and erosion. Tools like BRDF-Shop [CPK06] can be used to design user-defined materials. In addition, 3D-painting [HH90] allows a direct manipulation of the color on the 3D-surface. However, a direct design of the aging process is not yet possible.

3. Overview

We simulate the weathering effects with the help of gammatons introduced by Chen et al. [CXW*05], as they allow for effects resulting from interactions between objects, e.g. the dripping of patina or dirt, as shown in Fig. 2. In its essence, the idea is to emit particles from distant sources and to shoot them toward the objects. When a particle – called gammaton – hits a surface, an interaction is issued by applying rules depending on gammaton and surface properties. Thereby, the particle can deposit or take away material. Subsequently, the gammaton is either reflected, floats on the surface or is absorbed (see Fig. 3).

In our approach, all material, surface and normalization data is maintained in a texture atlas, which we refer to as *material atlas* (see Fig. 4). Originally, Chen et al. [CXW*05] proposed a surfel representation embedded in a k-d tree instead. But, not only do textures better fit into content pipelines, localization of deposited material by a texture lookup is naturally faster than a range query in a k-d tree.

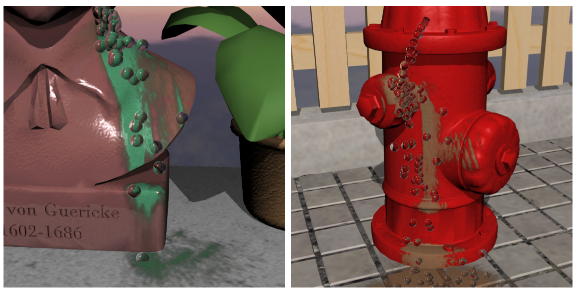


Figure 2: *Patina and dirt being transported by gammatons.*

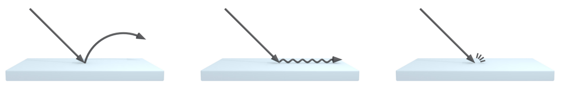


Figure 3: *Depiction of the three collision responses.*

The pipeline stages of the simulation are illustrated in Fig. 5 and are outlined in the following. The first action in the main loop is the invocation of the *simulation step*, which includes the emission of new gammatons, the iterative tracing of already existing gammatons and the detection of collisions with the surface geometry (elaborated in Section 4.1). In the next two stages, the *surface update* and the *gammaton update*, the material transport between the gammatons and the hit surfaces is issued (more detailed in Section 4.2). Furthermore, we avail Russian Roulette to determine the subsequent behavior of the particles (e.g. float, bounce or absorb) and specify their velocity for the next iteration. In the following stage, the *aging process*, we treat the interaction of materials over time, e.g. rusting in the presence of water and metal. Therefore, we apply a set of rules to the materials on the surface, further explained in Section 4.3. Based on the amount of each material stored in the *material atlas* (Section 4.4) we estimate new surface parameters in the *composition stage* (Section 4.5). Here, texture maps for the aged color, specular amount, normal and height are generated. These maps are used for the preview rendering (see Section 6) and can also be exported (alongside with the material atlas) for external use in game engines, DCC tools or serve as input to other rendering applications. Eventually, the pipeline starts anew with the next simulation step.

4. Simulation

The simulation involves the tracing of gammatons in the scene, as well as the interaction with the materials encoun-

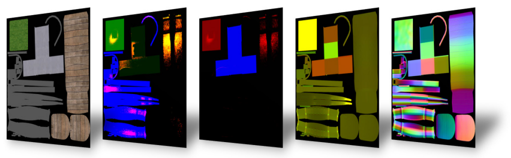


Figure 4: *The material atlas contains all surface information. In this image a subset of the atlas textures is shown. From left to right: original colors, amount of the respective material types (two images), texel-to-world scale and surface normals.*

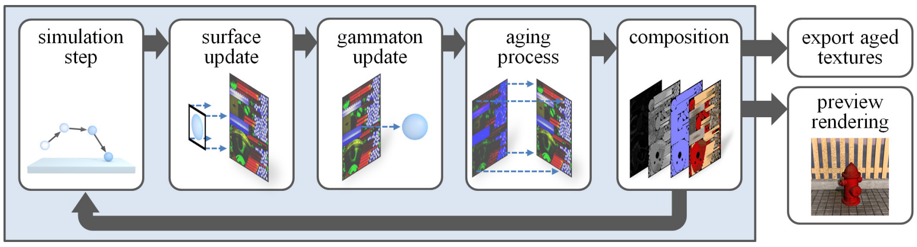


Figure 5: *Illustration of the simulation pipeline.*

tered on surfaces. We draw on Nvidia's GPU ray tracing API OptiX [PBD*10] for the tracing of gammatons and conduct the material updates and collision responses on the Direct3D side. For consistency, we process all material-related steps on the Direct3D side alone.

4.1. Tracing of Gammatons

Gammatons are launched in OptiX by so called entry programs. We launch the same entry program subsequently for each gammaton source, using parameters to define customized behavior, e.g. position, direction, emission speed and distribution. For the parallel processing of all gammatons later on, we assign each source an exclusive range of memory of a single gammaton stream shared by all sources. Before starting the gammaton tracing, all gammatons – associated to the source that launches – are either alive or ready to be spawned anew. A gammaton is considered alive if it is either still in mid-air or received a new direction in response to a collision. Alive gammatons are launched in their given direction from their last known position. All other gammatons either came to a stop on a nearly horizontal plane or left the scene and can thus respawn. By using OptiX, we trace the gammaton trajectories as a sequence of linear steps, each reporting a miss or hit. Misses are handled by adding gravity, followed by a recursive launch until the maximum recurrence depth is reached. Until then, a gammaton hits a surface, leaves the scene or remains in mid-air to be continued in the next simulation iteration. If a surface got hit, the gravity is applied accordingly to the traveled distance, only acting tangentially in case the gammaton is in the floating state. For gammatons in mid-air the current position is stored and for alive gammatons that hit a surface, a numerical offset ε in normal direction is added to the stored position. This offset will affect the speed of floating particles, as we will explain later on. When the tracing has finished, gammatons are in one of four possible states. They can still be mid-air, have hit a surface, have left the scene or be extinguished due to close-to-zero speed. Only gammatons that hit a surface pick up material, as described in Section 4.2, and respond to collisions by executing Russian Roulette on the events bouncing, floating and absorbing (see Fig. 3), according to their associated probabilities. A bounce is a reflection on the tangent plane given by the material-dependent estimated surface normal (see Section 4.5), randomized by a Phong lobe sampling, see Fig. 6. In order to avoid penetration of objects, i.e. shooting below the tangential plane of the geometric normal, rejection sampling is carried out. Floating particles are handled equally with the addition that the resulting direction is projected back to the tangential plane. Since we applied a numerical offset ε to the gammaton's position, a floating gammaton virtually hovers above the surface. We pull its direction back in the negated normal direction by the amount h to let the next ray aim back at the surface as shown in Fig. 7. Both the numerical offset ε as well as h steer the speed of floating gammatons. Their ability to grab for overhangs, when flowing bottom up, additionally depends on the range of the next ray, which is extended to accommodate the numerical offset from the surface. If the ray intersects, the gammaton reports a hit as usual; otherwise its direction is set to the gravity direction, letting the gammaton fall down. Moreover, the artist can decide whether gammatons slow down depending on their incoming angle (letting the resulting velocity only act tangentially), as it would be physically correct. In order to speed up the distribution it might be turned off. Aside from this, both the bouncing and floating gammatons are slowed down by a specific rate, approximating friction.

4.2. Material Transport

One of the fundamental mechanics of our approach is the transfer of material induced by flowing and bouncing gammatons. Whenever a gammaton hits the surface it picks up and deposits a certain amount of material. The initial material composition on the surfaces is assigned by material presets, e.g. stating that the surface on the grass mesh consists of 75% dirt and 25% organic. The quantity of the material

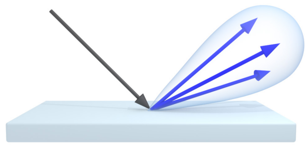
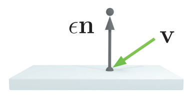
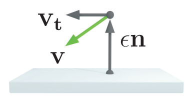


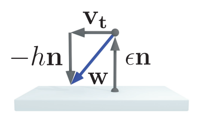
Figure 6: *For bounce events the reflected direction is randomized by a Phong lobe sampling. Arrows depict the incoming direction (●) and possible outgoing directions (●).*



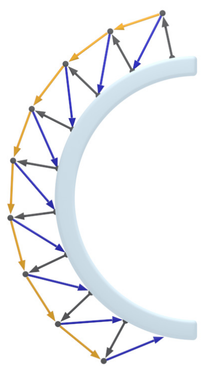
(a) Apply a numerical offset ε in normal direction to the hit position.



(b) Projecting the velocity $\mathbf{v}$ (●) into the tangential plane gives $\mathbf{v_t}$.



(c) Pulling of $\mathbf{v_t}$ in negated normal direction by the amount of h yields a new direction $\mathbf{w}$ (●).



(d) The visible course (●) of a gammaton floating around an overhang.

Figure 7: *Sequence of steps for handling a floating event.*

transferred depends on the amount of material lying on the surface and being carried by the gammatons. To incorporate the varying erosion speeds of the different material types, we introduced factors c_i to control the transferred amount for each material type i, e.g. to accommodate that water is more volatile than stone. The rates are defined once prior to the simulation and can be reused in other scenes. The only degree of freedom during the simulation is the ratio r between the amount of material carried away from the surface and added to the gammaton (and vice versa). Naturally, the overall material is preserved if this ratio is 1:1. Though to grant artists more freedom, we expose this parameter to the user, thereby allowing the artist not only to speed up the aging process, but also to reverse it, i.e. when gammatons pick up more material than they deposit, they ultimately wash away the material, see Fig. 8. With $m_i(\mathbf{x},t)$ being the material on a surface at the texture atlas location $\mathbf{x}$ at time t and $\gamma_i(\mathbf{x},t)$ being the material carried by a gammaton that hit at the texture atlas location $\mathbf{x}$ at time t, the following two linear ODEs can be deduced to describe the material transport. Note that

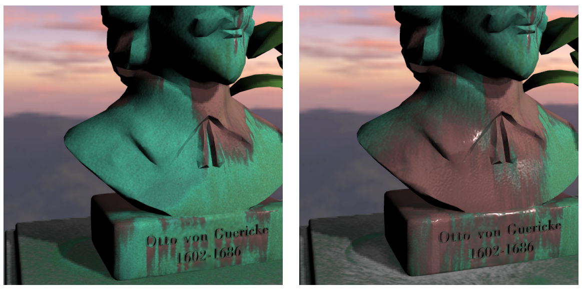


Figure 8: *Picking up more material than desspositing allows for washing away material (left: before, right:after).*

i enumerates the material types (water, dirt, ...).

$$\begin{aligned}\forall i: \frac{\mathrm{d}\gamma_i(\mathbf{x},t)}{\mathrm{d}t} &= (m_i(\mathbf{x},t)(1-r)-\gamma_i(\mathbf{x},t)\,r)\;c_i\\ \forall i: \frac{\mathrm{d}m_i(\mathbf{x},t)}{\mathrm{d}t} &= (-m_i(\mathbf{x},t)(1-r)+\gamma_i(\mathbf{x},t)\,r)\;c_i\end{aligned}$$

Thereby, a ratio r of zero means that material is only picked up and nothing is dropped, whereas a ratio of one means the opposite.

The material transport is invoked in pixel shaders by splatting small quads into the material atlas at the collision positions. The material deposited by the gammaton is added, whereas the material picked up is subtracted. The material, carried by the respective gammaton itself, is altered accordingly in the *gammaton update* stage, for which geometry shaders and transform feedback are utilized. Here, deposited material is subtracted and incidentally, picked up material is added to the gammatons.

An issue to deal with is the varying scale of the atlas textures. To increase the detail, usually more amount of texture area is spent on the most important objects. The remaining objects share the residual space, yielding a varying area an atlas texel covers in world space. As we issue material transfer whenever a gammaton hits a surface, the splat size of the material footprint must be adapted accordingly to the world space scale of the texel. Thus, we keep a normalizing, precomputed *texel-to-world scale* in the material atlas.

4.3. Aging Rules

We directed our focus on the design of a flexible rule system used to guide the aging processes. To facilitate a variety of aging phenomena in a broad scope of multifaceted scenes we propose an asset-like system to define – and save – customized rules, as inspired by material libraries nowadays found in almost every DCC program. Such a system can be implemented elegantly and efficiently by using the dynamic shader linkage. This concept has been long known in Nvidia's shader language CG, thus allows targeting older hardware. Most recently, it found its way into the graphics APIs DirectX 11 and OpenGL 4.0 – in the latter it is known as subroutine functions.

We model a rule as an implementation of an interface defining a method that alters a given material property configuration. This way, we allow for potentially complex and sophisticated rules. In the following, we explain the three rules used throughout the paper. For each rule, we define an α-parameter that scales the amount of material generated or removed.

Rust. The first rule turns metal into rust in the presence of water:

$$\frac{\mathrm{d}m_{rust}(\mathbf{x},t)}{\mathrm{d}t} = min(m_{water}(\mathbf{x},t), m_{metal}(\mathbf{x},t))\,\alpha_{rust}.$$

Decay. The second rule is used to weather wood by producing organic material and dirt:

$$\begin{aligned}\frac{\mathrm{d}m_{organic}(\mathbf{x},t)}{\mathrm{d}t} &= min(m_{water}(\mathbf{x},t), m_{wood}(\mathbf{x},t))\,\alpha_{organic}\\ \frac{\mathrm{d}m_{dirt}(\mathbf{x},t)}{\mathrm{d}t} &= min(m_{water}(\mathbf{x},t), m_{wood}(\mathbf{x},t))\,\alpha_{dirt}.\end{aligned}$$

Evaporation. The last rule evaporates water over time:

$$\frac{\mathrm{d}m_{water}(\mathbf{x},t)}{\mathrm{d}t} = -\alpha_{water}.$$

4.4. Data Formats

In the following section, we take a closer look on the data structures involved in the implementation. Of particular interest are the formats used to store the materials on the surface and at the gammatons. For the information on the surfaces we use the material atlas, thus beforehand a parameterization is required, which we obtained semi-automatically using standard DCC tools. The material atlas contains colors + specular coefficient (RGBA8), face normals (RGB16F), face tangents (RGB16F), shading normals (RGB16F), materials (ping-pong of 2× RGBA8), original materials (2× RGBA8) and a texture-to-world scale (RG16F). The memory consumption is a non-negligible issue, especially due to the high bandwidth workload, thus we reduced the materials to 256 discrete steps each, using 8 materials in total. To attain more granularity on the lowly discretized materials, we invoke the aging rules with a customizable probability. As opposed to our probabilistic approach, it suggests itself to invoke a rule every n frames, which however is not viable as it may yield discernible, periodic popping artifacts in the rendered image.

The 8 material slots can be assigned to certain materials, as they are required in the particular scene. Throughout the paper we needed 7 slots for water, dirt, metal, wood, organic, rust and stone. Output of the simulation are the composed textures, containing the aged color (RGB8), normal (RGB8) height (R8) and specularity (R8). Each gammaton stores its position (float3), the velocity (float3), the carried material (uint2), the texture coordinates (float2) of the surface it last hit, its state (uint) – as explained in Section 4.1 – and a seed for the linear congruence random number generator (uint).

4.5. Composition

The last stage of the pipeline is the composition, which is responsible for the estimation of the new surface properties, based on the current amount of the different material types. More specifically, we estimate the aged diffuse color, specular values, tangent space normals and a height map for displacement mapping (see Fig. 9(a)). Note that the diffuse color and specular values are used for the Phong reflection model of the preview renderer and may directly be used in the final renderer as diffuse and specular maps as in Fig. 1. Our composition does not place a limit for photo-realistic content production, since sequent DCC tools may also import the material atlas to compose input for more complex reflection models, thereby still having full flexibility.

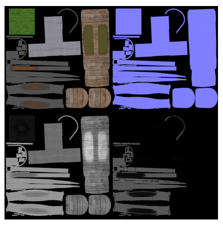
The diffuse color of the aged material is estimated by computing for each material type the difference between the current material in the atlas and the original material at the regarded position. This difference directly correlates with the influence of the material type on the resulting color. The color values $\mathbf{s}$, representing the material types, are fetched from textures referenced in the aforementioned material presets (see Fig. 9(b)). To grant possibilities for adjusting the result during or after the simulation, two degrees of freedom are introduced per material type: one for controlling the base strength b and one for introducing a random variation v – again both pre-defined by presets. This enables the artist to attain various appearances e.g. a homogenous patina of copper or noisy rust stains on iron surfaces. Let ξ denote a uniformly destributed random number. By estimating a color $\mathbf{c}_{deposit}$ for the material deposited on the surface:

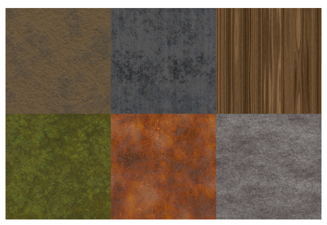
$$\mathbf{c}_{deposit} = \sum_i \max(0, m_i(\mathbf{x},t) - m_i(\mathbf{x},0)) \cdot \mathbf{s}_i \cdot (b_i + v_i \cdot \xi_i)$$

the resulting diffuse color is formally defined as

$$\mathbf{c}_{diffuse} = (1 - \sum_i (b_i + v_i \cdot \xi_i))\,\mathbf{c}_{org} + \sum_i (b_i + v_i \cdot \xi_i)\,\mathbf{c}_{deposit}.$$

The weight $\sum(b_i + v_i \cdot \xi_i)$ blends between the original color $\mathbf{c}_{org}$ and the deposited color $\mathbf{c}_{deposit}$. The specular values are estimated similarly, leaving out the color textures. Aside


(a) The composed textures aged color, normal, height and specularity.


(b) Textures associated to the materials dirt, iron, wood, grass, rust and stone.

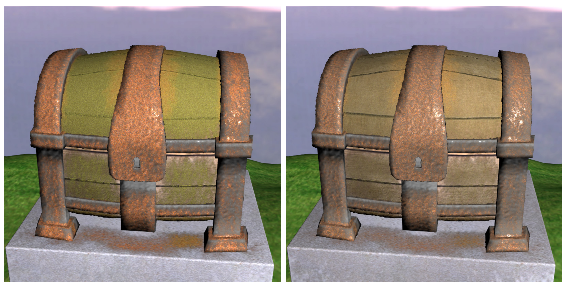

(c) Two differently composed aged chests.

Figure 9: *Texture composition in the Chest scene.*

from the amounts of the different material types, one byte is reserved to store the overall material height, which is initialized with values from input height maps. In the composition stage of the pipeline, the height is influenced by the amount of material on the surface, e.g by rust and organic, resulting in an output height map. Finally, we use the height map to calculate tangent space normals.

5. User Interaction

The following section outlines the typical user interactions undertaken during usage of our program. For convenience, our system allows the user to rearrange both the scene objects (move, rotate and scale) and the gammaton emitters. Inspired from user interface concepts often found in sculpting tools, we additionally allow the artist to paint on surfaces by placing the emitter at the selected location with an offset in normal direction, adjustable by scrolling the mouse wheel. Furthermore, the variance from the normal direction (concentrated beam vs. drizzle) and the size of the emitter can be adjusted.

To decrease the number of user parameters, we steer the aforementioned Russian Roulette probabilities used to determine the behavior (float, bounce or absorb), and the deceleration rate of the gammatons with one slider, as visualized in Fig. 10. On one extreme all gammatons bounce without loss of speed. Moving the slider up to 1/3 continuously decelerates the gammatons. Going from 1/3 to 2/3 tilts the ratio to solely floating behavior, whereas the last third steers the deceleration of the floating gammatons. The curves for the deceleration rates were chosen reasonably (other choices are imaginable) and are – for simplicity on the user's end – assumed to be constant for the entire scene. It is a possible future work to investigate how much artists steer against the curves when the rates are derived from material properties. After adjusting the scene-dependent numerical parameters, the movement simulation can be steered by this single slider. In conjunction with the ratio between depositing and picking up of material, the course of the simulation is mainly steered by only two independent parameters. After (or even during) the simulation, post-controls can be used to adjust the overall strength of the weathering for each type of effect (see Fig. 9(c)), which modifies the blending weights, applied in the composition of the output textures, see Section 4.5.

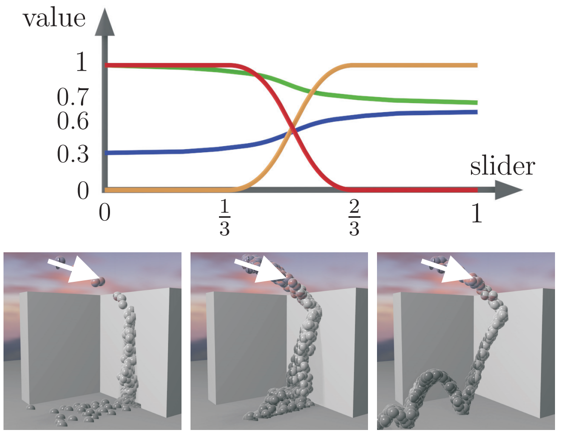


Figure 10: *A single slider steers the rate of deceleration for bouncing (●) and floating (●), and the probabilities of bouncing (●) and floating (●). The figures (a) to (c) show several choices for the bounce behavior.*

6. Display

An adequate intermediate output of the current simulation results is of great importance, as it bears upon the artist's ability to effectively steer the simulation towards the aimed look, thus must also be coherent with the images generated by the renderer that takes the composed textures (and optionally the material atlas) as input. For this reason, an adaptation to the employed production pipeline is necessary in order to resemble the appearance attained by the final renderer (either in film production or games).

For our test cases, we render the scene with conventional forward rendering, thereby binding the output textures from the composition stage, i.e. the aged color, normals, height and specularity. We avail tessellation shaders for the displacement (using Phong tessellation by Boubekeur and Alexa [BA08] and a distance-dependent continuous level-of-detail) along with bump mapping to provide a general feedback for the quality of the normal and height map. We employed classic Phong shading, comprising the specularity and the aged color, as well as shadows casted from the main light. Additionally, we added ray-traced single-bounce reflections to display moisture. To enrich the weathered look, we additionally place plantlets in regions of high organic lichen, yielding a more detailed representation as can be seen in Fig. 11. Given a fixed number of plantlets to place in total, we randomly reseed plantlets to adapt to the current organic distribution in the scene.

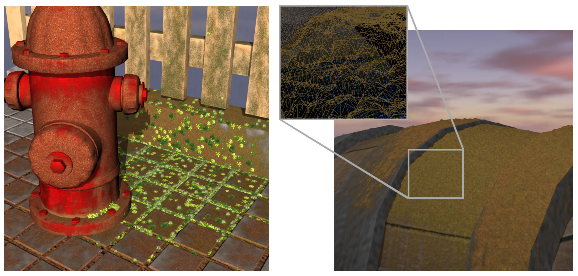


Figure 11: *Depictions of the intermediate output. Left: plantlets distributed in regions of high lichen growth. Right: Distance-dependent tessellation with displacement mapping, depending on the amount of rust.*

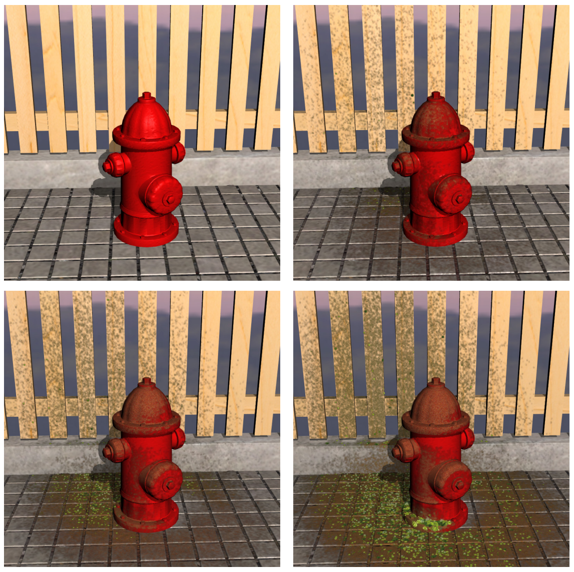


Figure 12: *Aging of a selected part of the hydrant scene, shown at different time steps.*

7. Results

Our main goal was to provide a fast aging simulation that allows artists to interactively steer and design the aging process. Therefore, our simulation presents an intermediate output at interactive rates and allows modifying (e.g. moving) of gammaton sources, giving the artists precise control. Fig. 12 shows the original scene and the aged scene at different time steps. In the content creation workflow, our simulation is one step among many. Figure 1 shows renderings done with our exported aged material data in standard DCC tools.

Since our approach is based on a texture atlas, seams at texture coordinate discontinuities are to expect if the splat size is bigger than one atlas texel. When normalizing the splat size to one texel, no seams occur as shown in Fig. 13.

The machine we experimented on is equipped with an Intel Core 2 Quad Q9650 CPU, an Nvidia GeForce GTX 560 Ti GPU with 3.5 GB VRAM and 4 GB RAM. The simulation is mainly limited by the memory and the associated bandwidth penalties. Table 1 shows the memory requirements for the material atlas at different atlas resolutions and Figure 14 compares the obtained visual quality. Note that the system is readily able to interactively simulate on an atlas resolution that yields production level quality, thus no further off-line high-quality simulation process is needed. The main bottleneck is the high transfer rate during the composition. The simulation of the gammatons, i.e. the OptiX side of the simulation, is very fast and perfectly scalable to the scene size. Table 2 shows the timing breakdown of the pipeline stages (5k gammatons per iteration, screen resolution of 800 × 800 pixel) and Table 3 shows the timings of the intermediate output dependend on the atlas resolutions. About 80% of

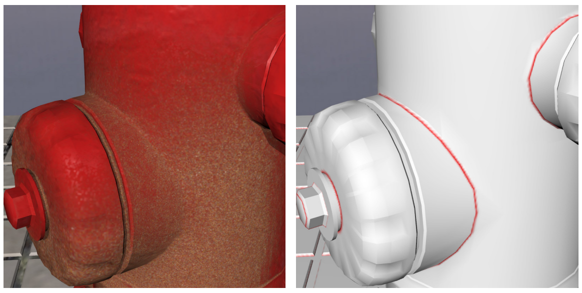


Figure 13: *With a splat size of one atlas texel no seams occur. Left: rusting fire hydrant, right: visualization of texture coordinate discontinuities.*

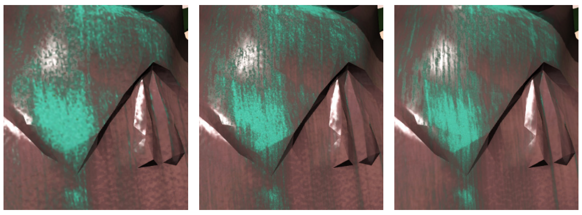


Figure 14: *Comparison of the impact of different atlas resolutions on the quality, from left to right: 1k×1k, 2k×2k, 4k×4k. Advancing to 4k×4k is not worthwhile in this scene.*

Table 1: *Memory requirements for various atlas sizes.*

Resolution	Material Atlas	Composed Textures
1k × 1k	56 MB	8 MB
2k × 2k	224 MB	32 MB
4k × 4k	896 MB	128 MB

Table 2: *Timing breakdown in ms for the pipeline stages at an atlas resolution of 1k × 1k.*

Step	Chest	Hydrant	Otto
Simulation Step	2.76	2.72	2.93
Surface Update	0.03	0.03	0.03
Gammaton Update	0.03	0.03	0.03
Aging Process	0.25	0.25	0.25
Composition	5.91	6.04	5.99
Preview Rendering	7.95	7.12	8.81
Total Time	16.93	17.02	18.04

Table 3: *Total timings in ms achieved at different atlas sizes.*

Resolution	Chest	Hydrant	Otto
1k × 1k	16.93	17.02	18.04
2k × 2k	36.26	37.82	36.58
4k × 4k	111.02	112.41	111.54

the preview rendering costs originate in the brute-force reflection rendering, which can be improved or avoided if not needed. Also note that the preview rendering and the composition – which are both the slowest components – are optional to the simulation. Breaking the composition down, to be carried out over a span of a few frames, and employing a more sophisticated and optimized renderer allows to adjust to the narrow time budget of real-time applications, e.g. modern games.

8. Conclusion and Future Work

In this paper, we presented the first interactive material aging simulation by tracing gammatons on the GPU. We employed a simple set of rules to achieve the most common aging effects (e.g. dirt, rust, organic and water precipitate) and displayed those in a few seconds in which the scene progressively and visibly ages. Additionally, we used user interaction techniques known from sculpting for the adding of filigree detail, thereby allowing to directly steer and design the aging process.

Yet open to further research is the simulation on large scenes. Currently, our approach is limited to a few objects (chosen by the artist as a region of interest), as the memory requirements for the material atlas limit the quality of the simulation. Possible out-of-core approaches to consider are presented by Lefebvre et al. [LDN04] and virtual texturing [vW09]. A number of extensions to our approach are imaginable. If more memory was available (e.g. by compressions), it would be possible to add multiple layers of material, not only one as we do now. Stack-based terrains [LMS11] are a possible source of inspiration. Accompanied with this is the gradual peeling of the layers, possibly initiating a more distinctive deformation of the surface, which could go beyond the capabilities of single-pass tessellation shaders. Another important step is the implementation of a more detailed temporal aging behavior, since many materials are subject to a non-linear aging process [GTR*06].

References

[BA08] Boubekeur T., Alexa M.: Phong tessellation. *ACM Transactions on Graphics 27*, 5 (Dec. 2008), 141:1–141:5. 6

[BPMG04] Bosch C., Pueyo X., Merillou S., Ghazanfarpour D.: A Physically-Based Model for Rendering Realistic Scratches. *Computer Graphics Forum 23*, 3 (2004), 361–370. 2

[CPK06] Colbert M., Pattanaik S., Krivanek J.: BRDF-shop: creating physically correct bidirectional reflectance distribution functions. *IEEE Computer Graphics and Applications 26*, 1 (2006), 30–36. 2

[CXW*05] Chen Y., Xia L., Wong T.-T., Tong X., Bao H., Guo B., Shum H.-Y.: Visual Simulation of Weathering by Gammaton Tracing. *ACM Transactions on Graphics 24* (2005), 1127–1133. 2

[DEJ*99] Dorsey J., Edelman A., Jensen H. W., Legakis J., Pedersen H. K. h.: Modeling and rendering of weathered stone. *ACM Transactions on Graphics*, Annual Conference Series (1999), 225–234. 2

[DGA04] Desbenoit B., Galin E., Akkouche S.: Simulating and modeling lichen growth. *Computer Graphics Forum 23*, 3 (2004), 341–350. 2

[DH96] Dorsey J., Hanrahan P.: Modeling and Rendering of Metallic Patinas. In *SIGGRAPH* (1996), vol. 30, ACM Press, pp. 387–396. 2

[DPH96] Dorsey J., Pedersen H. K. h., Hanrahan P.: Flow and Changes in Appearance. In *SIGGRAPH* (1996), vol. 30, ACM, pp. 411–420. 2

[GC01] Gobron S., Chiba N.: Crack pattern simulation based on 3D surface cellular automata. *The Visual Computer 17*, 5 (2001), 287–309. 2

[GTR*06] Gu J., Tu C., Ramamoorthi R., Belhumeur P., Matusik W., Nayar S.: Time-varying surface appearance: acquisition, modeling and rendering. *ACM Transactions on Graphics 25*, 3 (2006), 762–771. 2, 8

[HH90] Hanrahan P., Haeberli P.: Direct WYSIWYG painting and texturing on 3D shapes. *SIGGRAPH 24*, 4 (1990), 215–223. 2

[HTK98] Hirota K., Tanoue Y., Kaneko T.: Generation of crack patterns with a physical model. *The Visual Computer 14* (1998), 126–187. 2

[HTK00] Hirota K., Tanoue Y., Kaneko T.: Simulation of three-dimensional cracks. *The Visual Computer 16*, 7 (2000), 371–378. 2

[KRB11] Kider J. T., Raja S., Badler N. I.: Fruit Senescence and Decay Simulation. *Computer Graphics Forum 30*, 2 (2011), 257–266. 2

[LDN04] Lefebvre S., Darbon J., Neyret F.: *Unified Texture Management for Arbitrary Meshes*. Rapport de recherche RR-5210, INRIA, 2004. 8

[LMS11] Löffler F., Müller A., Schumann H.: Real-time Rendering of Stack-based Terrains. In *VMV* (2011), pp. 161–168. 8

[MDG01] Merillou S., Dischler J.-M., Ghazanfarpour D.: Corrosion: Simulating and Rendering. In *Graphics Interface* (2001). 2

[MG08] Merillou S., Ghazanfarpour D.: A survey of aging and weathering phenomena in computer graphics. *Computers & Graphics 32*, 2 (2008), 159–174. 2

[PBD*10] Parker S. G., Bigler J., Dietrich A., Friedrich H., Hoberock J., Luebke D., McAllister D., Stich M.: OptiX : A General Purpose Ray Tracing Engine. *ACM Transactions on Graphics 29*, 4 (2010), 1–13. 3

[PPD01] Paquette E., Poulin P., Drettakis G.: Surface Aging by Impacts. *Graphics Interface* (2001), 175–182. 2

[PPD02] Paquette E., Poulin P., Drettakis G.: The Simulation of Paint Cracking and Peeling. *Graphics Interface* (2002), 59–68. 2

[vW09] van Waveren J.: id tech 5 challenges - from texture virtualization to massive parallelization. In *SIGGRAPH 2009 Beyond Programmable Shading course* (2009). 8

[WNH97] Wong T.-T., Ng W.-Y., Heng P.-A.: A Geometry Dependent Texture Generation Framework for Simulating Surface Imperfections. In *Eurographics Rendering Workshop 1997* (1997), Springer-Verlag, pp. 139–150. 2

[WTL*06] Wang J., Tong X., Lin S., Pan M., Wang C., Bao H., Guo B., Shum H.-Y.: Appearance manifolds for modeling time-variant appearance of materials. *ACM Transactions on Graphics 25*, 3 (2006), 754. 2